

基于 SQL 语法树的 SQL 注入过滤方法研究

韩宸望^{1,2}, 林晖^{1,2}, 黄川^{1,2}

(1. 福建师范大学数学与计算机科学学院, 福建 福州 350117;

2. 福建师范大学福建省网络安全与密码技术重点实验室, 福建 福州 350117)

摘要: Web 应用的发展, 使其涉及的领域也越来越广。随之而来的安全问题也越来越严重, 尤其是 SQL 注入攻击, 给 Web 应用安全带来了巨大的挑战。针对 SQL 注入攻击, 将基于 SQL 语法树比较的安全策略引入用户输入过滤的设计中, 提出了一种新的 SQL 注入过滤方法。实验结果表明, 该方法能够有效地防止 SQL 注入攻击, 并有较高的拦截率和较低的误报率。

关键词: SQL 注入攻击; Web 安全; SQL 语法树; 用户输入过滤

中图分类号: TP309

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2016.00113

Research on the SQL injection filtering based on SQL syntax tree

HAN Chen-wang^{1,2}, LIN Hui^{1,2}, HUANG Chuan^{1,2}

(1. School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350117, China;

2. Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou 350117, China)

Abstract: The development of Web application make its areas become more and more widely. Followed by a security problem is becoming more and more serious, especially for the SQL injection attacks, which bring a huge challenge to the Web application security. A new SQL injection filtering method was proposed to detect SQL injection attack by introducing a security strategy based on SQL syntax tree to the design of the user input filtering. The experimental results show that the method can effectively prevent SQL injection attacks, and has higher recognition rate and lower rate of false positives.

Key words: SQL injection attack, Web security, SQL syntax tree, user input filtering

1 引言

随着互联网技术的快速发展, Web 技术在许多领域都得到了广泛的应用, 且已成为世界经济发展中的关键技术之一^[1]。然而, 随着 Web 技术各个领域的普及和发展, 也面临着越来越多

的安全威胁。并且, Web 安全已经成为网络安全最主要的来源之一, 因此, 对 Web 安全的研究也就成为网络安全领域的一个重点和热点^[2]。

目前, 影响 Web 安全的因素众多, 其中, SQL 注入是最高危的漏洞之一。在 2008 年~2010 年, SQL 注入攻击连续 3 年在开放式 Web 应用程序安

收稿日期: 2016-08-17; 修回日期: 2016-10-14。通信作者: 林晖, linhui@fjnu.edu.cn

基金项目: 国家自然科学基金资助项目 (No.61363068, No.61472083); 福建省引导基金资助项目 (No.2016Y0031); 福州市科技局基金资助项目 (No.2015-G-54, No.2015-G-84)

Foundation Items: The National Natural Science Foundation of China (No.61363068, No.61472083), Pilot Project of Fujian Province (No.2016Y0031), Project of Fuzhou Municipal Science and Technology Bureau (No.2015-G-54, No.2015-G-84)

全项目（OWASP，Open Web Application Security Project）年度十大攻击行为中排名第一。在 2011 年~2015 年，SQL 注入攻击连续 5 年排入 OWASP 年度十大攻击行为。

基于上述分析，本文有机地融合了用户输入过滤措施和 SQL 语法树结构比较方法，提出了一种基于 SQL 语法树的 SQL 注入过滤方法。实验结果表明，该方法可以有效地防止攻击者绕过防注入检测，提高了对 SQL 注入攻击的拦截；此外，在有效拦截 SQL 注入攻击的同时，该方法还降低了过滤用户输入数据的误报率。

2 SQL 注入攻击与防注入检测技术

2.1 SQL 注入攻击定义

SQL 注入攻击是一种由攻击者通过影响应用程序向后台数据库传递的 SQL 查询而引发的攻击。文献[3]中将 SQL 注入攻击定义为：攻击者在应用程序的查询语句中插入恶意指令来获取数据，即攻击者利用现有应用程序，将包含恶意代码的数据传递给数据库，从而修改 SQL 自身的语法和功能，导致了恶意的 SQL 语句被 SQL 解释器执行而产生攻击。

2.2 SQL 注入攻击原理

SQL 注入攻击是一种通过操纵 Web 输入来修改后台 SQL 语句以利用代码进行攻击的技术[4]。SQL 注入攻击种类很多，不同种类的攻击方式虽然有所区别，但是原理相同：即把通过在 Web 输入中包含恶意代码，使 Web 应用程序将恶意代码拼接到动态 SQL 语句中，从而获取信息。

下面通过一个万能账号的案例来说明 SQL 注入攻击。例如，一个登录页面，用户输入正确的账号和密码后，Web 程序会查询数据库，如果存在此用户并且密码正确，将成功登录；如果用户不存在或密码不正确，则会提示账号或密码错误。其中，该数据库为 MySQL 数据库，Users 表中有 3 列，分别为：user name、password、type 这种场景中，该登录页面的动态 SQL 语句为

```
select * from users where username='"+user
name +"' and password='"+ password +'"'
```

攻击者将使用账号“'or 1=1 --”登录，密码为“123456”。此时，SQL 语句变为

```
select * from users where user name ='1'or
'1'='1'--and password ='123456'
```

其中，'1'='1'为永真，“--”为注释符，注释符后面的 SQL 语句不执行。

经上述分析，Web 应用程序执行的 SQL 语句恒成立，因此，攻击者能够在不知道账号和密码的情况下登录系统。

2.3 防注入检测技术

Web 应用通常会使用部署在应用代码中或以 WAF、IDS 形式存在的输入过滤器来防御 SQL 注入攻击[5]。下面用以上文案例中的环境来介绍大小写、SQL 注释、编码和动态查询等防注入检测绕过技术。

1) 大小写技术

因为数据库使用不区分大小写的方式处理 SQL 关键字，因此，大小写技术通过变换攻击字符串中字符的大小写来绕过防注入检测。例如，如果下列输入被阻止：'UNION SELECT 1, password,3 FROM users--。可以通过下列方法来绕过防注入检测：'UnIoN SelEcT 1, password,3 fRom users --。

2) SQL 注释技术

使用内联注释来创建 SQL 代码段能够绕过多种输入防注入检测，尤其是简单的模式匹配防注入检测。攻击者可以在过滤了空白符的情况下使用内联注释来分隔每个关键字。如

```
'/**/**/UNION/**/**/SELECT/**/**/1,pass
word,3/**/**/FROM /**/**/account/**/--
```

在较早的 MySQL 版本中，甚至可以在 SQL 关键字内部使用内联注释，而绕过防注入检测。如

```
'/**/**/UN/**/I/**/ON/**/**/S/**/EL/**/E/*
*/CT/**/**/1,password,3/**/**/FR/**/O/**/M/**/
/**/users/**/--
```

3) 编码技术

攻击者可以通过编码来避免恶意代码中的关键字被黑名单匹配到，从而绕过防注入检测。编码技术有非常多种，如 URL 编码、UTF-8、十六进制编码。其中，攻击中最为常用的编码技术为 URL 编码。假设过滤器能够阻止空白符和内联注释符/*，但无法阻止用了 URL 编码的内联注释的 SQL 攻击。在这种情况下，可以用以下语句来绕

过防注入检测

```
%2f%2a*/UNION%2f%2a*/SELECT%2f%2a*/password%2f%2a*/FROM%2f%2a*/users%--
```

4) 动态查询技术

由于许多数据库都允许动态执行 SQL 查询,导致攻击者可以使用动态查询来绕过防注入检测。不同的数据库中动态查询执行的实现会有所不同。例如,不同数据库使用不同的语法来拼接字符串。假设 SQL 关键字 SELECT 被过滤了,可以用以下方式构造它。

Oracle: 'SEL' || 'ECT'

SQL Server: 'SEL' + 'ECT'

MySQL: 'SEL'ECT'

也可以使用 CHAR 函数 (Oracle 中为 CHR 函数) 来构造 SELECT

```
CHAR(83)+CHAR(69)+CHAR(76)+CHAR(69)+CHAR(67)+CHAR(84)
```

3 相关工作

近年来,国内外对 SQL 注入的研究取得了不少成果。在一定程度上防止了 SQL 注入攻击。这些成果大致分为以下几类。

1) 用户输入过滤

用户输入过滤方法通过对用户提交的数据采用关键字匹配,基于正则表达式的规则匹配等方法进行过滤。用户输入过滤能够有效防御已知的 SQL 注入攻击,但无法对未知的 SQL 注入攻击进行有效的防御。传统的用户输入过滤方案有关键字过滤和基于正则表达式的特征规则过滤。

Ivan Ristic 开发的开源项目 ModSecurity^[6]和 Roesch M 开发的入侵检测软件 Snort^[7],这2个项目都是通过制定攻击的特征规则集过滤用户输入,从而防御 SQL 注入攻击。

关键字过滤通过过滤 SELECT、OR、AND 等关键字防御 SQL 注入,但是对 SQL 注入的拦截率有限和较高的误报率。基于正则表达式的特征规则过滤相对于关键字过滤有效地提高了率,降低了误报率,能够有效防御 SQL 注入。

2) 基于机器学习的 SQL 注入检测

基于机器学习的 SQL 注入检测技术利用机器学习算法学习所有可能的 SQL 注入攻击,但是

该方法的效果十分依赖于机器学习算法的精确度和学习样本的准确度。Kim 等^[8]提出了一种基于数据挖掘和机器学习的 SQL 注入攻击检测方法。该方法提取了数据库日志中内部查询树的特征向量,之后通过 SVM 算法对这些特征向量进行分类,从而达到防御 SQL 注入攻击的目的。

3) 语法结构比较

语法结构比较的方法能够高效地防御 SQL 注入攻击,但是它的覆盖率很难达到 100%,并且这种方法可能需要修改源码。石聪聪等^[9]提出了一种基于 SQL 语法树特征匹配的方法,该方法在后台中维持一个由正常 SQL 语句提取的特征知识库。当 Web 应用程序生成动态 SQL 语句时和后台的知识库进行匹配。若匹配成功,则为合法 SQL 语句;若匹配不成功,根据基于风险值的动态特征过滤算法来进行识别。通过分析可知,该方法在知识库中的语法树特征数量较大时,会存在使匹配效率降低的问题。

4) 参数化查询和存储化过程^[10,11]

参数化查询是在代码层对 SQL 注入攻击进行防御^[12]。现今的大部分编程语言都能通过参数化查询语句访问数据库。参数化查询语句让数据库访问 API 可以使用占位符或绑定变量来向 SQL 查询提供参数。使用参数化查询能够避免或解决很多应用中常见 SQL 注入问题。

正确地配置存储过程的访问权限能够有效地防止或减轻 SQL 注入攻击产生的影响,提高系统的安全性^[13]。并且大部分数据库现在都能够支持存储过程,允许通过存储过程对数据库进行访问。

5) 敏感信息加密

对数据库中的账号密码、客户信息、交易信息等敏感数据进行加密,能够尽量地降低受到 SQL 注入攻击后的危害。杨玉龙等^[14]提出了一种新的防止 SQL 注入的方案,结合同态加密技术,对用户的输入进行同态加密,使攻击者输入的特殊字符或拼接的 SQL 语句失去其本身的意义,该方案的使用有效地防止了 SQL 注入攻击。

6) 网络测量

SQL 注入在请求长度、连接数以及特征串等方面相较于正常用户请求的流量有较大区别。据此,赵宇飞等^[15]首次从网络流量分析的角度检测 SQL

注入行为, 提出了 LFF (length-frequency- feature) 检测方法, 通过连接频率检测、长度检测和特征串检测, 能够较准确地发现网络中的 SQL 攻击行为。

4 基于 SQL 语法树的 SQL 注入过滤方法

4.1 概述

对于正常的用户输入数据, 根据数字型和字符型, 将用户输入的数据代入 SQL 语句 “select a from b where c=?” (其中? 为占位符)。其生成的 SQL 语句的特征都是相同的, 如该 SQL 语法树的节点总数是一样的。以用户输入 admin 为例, 其 SQL 语法树的结构如图 1 所示。

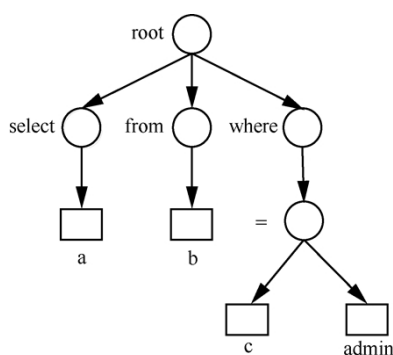


图 1 正常的 SQL 语法树

而对于 SQL 注入攻击输入, 将攻击输入数据代入相同的 SQL 语句中将会报错或改变原有的 SQL 语句的语法结构, 如增加某个节点、语法树结构变化等。以用户输入 admin' or '1'='1 为例, 其 SQL 语法树的结构如图 2 所示。

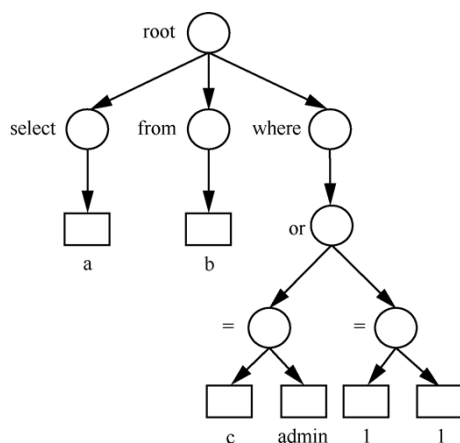


图 2 注入攻击的 SQL 语法树

因此, 如果某个输入数据带入 SQL 语句 “select a from b where c=?” (? 为占位符) 后, 该 SQL 语句

的语法结构特征 (SQL 语法树节点总数) 与预期的不符, 则说明该用户输入可能为 SQL 注入攻击。

基于上述分析, 本文结合了用户输入过滤措施和 SQL 语法树结构比较措施的特点, 提出了一种基于 SQL 语法树的 SQL 注入过滤方法。该方法的设计思路为: 将 SQL 注入过滤分为转码阶段和 SQL 语法树比较阶段这 2 个部分。其中, 转码阶段检测用户输入的数据, 将用户输入中字符的大小写、编码、注释符进行转换, 以防止攻击者绕过检测。SQL 语法树比较阶段通过将用户输入数据分别以数字型和字符型带入 “select a from b where c=?” (? 为占位符) 中, 分别对两条生成后的 SQL 语句进行语法解析生成语法树。若 2 棵 SQL 语法树的特征 (SQL 语法的节点总数) 跟期望中的特征一样, 则该用户输入是合法的。若两棵 SQL 语法树其中一颗 SQL 语法树的特征与期望中的不一致, 则判断该用户输入为 SQL 注入攻击。

4.2 SQL 注入攻击检测过程

本文提出的基于 SQL 语法树的 SQL 注入过滤方法的详细流程如图 3 所示。

该方法的具体工作过程描述如下。

1) 首先, 将用户输入中字符的大小写、编码、注释符进行转换。消除 SQL 语句前后的空白, 将其中的连续空白字符 (包括空格、TAB 和回车换行) 替换成单个空格。之后将 SQL 语句全变成小写形式 (或大写形式), 并转化各种编码, 以此防止攻击者绕过检测。

2) 在不知道用户输入的该参数是字符型还是数字型的情况下。先假设用户输入的该参数为字符型, 用输入的数据 X 代入 SQL 语句 “select a from b where c=?” (? 为占位符), 得到 SQL 语句 “select a from b where c=X”。

3) 对 SQL 语句 “select a from b where c=X” 进行 SQL 解析得到 SQL 语法树。若解析失败, 则直接结束, 并将该输入判定为 SQL 注入; 若解析成功继续步骤 4)。

4) 将步骤 3) 生成的 SQL 语法树的特征 (SQL 语法的节点总数) 与期望中的特征进行匹配。若一样, 继续步骤 5); 若不一样直接结束, 并将该输入判定为 SQL 注入攻击。

5) 判断用户输入的数据中第一个空格前的数

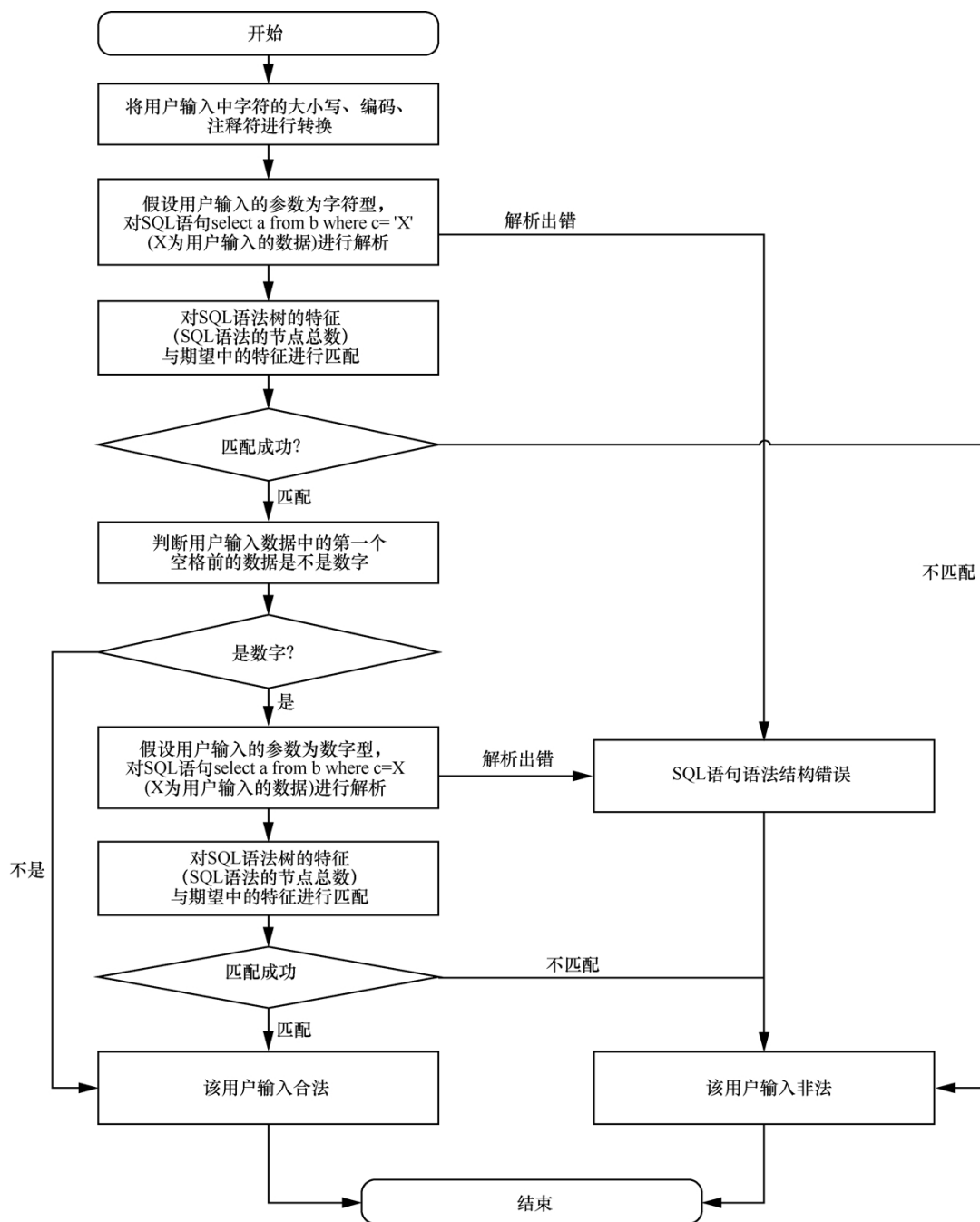


图3 过滤流程

据是不是数字。若是数字，继续步骤6)；若不是数字，直接结束，并将该输入判定为正常输入。

6) 假设用户输入的该参数为数字型，将用输入的数据 X 代入 SQL 语句“select a from b where c=?”（?为占位符），得到 SQL 语句“select a from b where c=X”。

7) 对 SQL 语句“select a from b where c=X”进行 SQL 解析得到 SQL 语法树。若解析失败，

则直接结束，并将该输入判定为正常输入；若解析成功，继续步骤8)。

8) 将步骤7)生成的 SQL 语法树的特征(SQL 语法的节点总数)与期望中的特征进行匹配。若一样，则将该输入判定为正常输入；若不一样，则将该输入判定为 SQL 注入攻击。

4.3 模块介绍

本文所提方法的设计模块如图4所示。